

Context Forest for efficient object detection with large mixture models

Davide Modolo
University of Edinburgh
d.modolo@sms.ed.ac.uk

Alexander Vezhnevets
University of Edinburgh
avezhnev@inf.ed.ac.uk

Vittorio Ferrari
University of Edinburgh
vferrari@staffmail.ed.ac.uk

Abstract

We present Context Forest (ConF) — a technique for predicting properties of the objects in an image based on its global appearance. Compared to standard nearest-neighbour techniques, ConF is more accurate, fast and memory efficient. We train ConF to predict which aspects of an object class are likely to appear in a given image (e.g. which viewpoint). This enables to speed-up multi-component object detectors, by automatically selecting the most relevant components to run on that image. This is particularly useful for detectors trained from large datasets, which typically need many components to fully absorb the data and reach their peak performance. ConF provides a speed-up of 2x for the DPM detector [1] and of 10x for the EE-SVM detector [2]. To show ConF's generality, we also train it to predict at which locations objects are likely to appear in an image. Incorporating this information in the detector score improves mAP performance by about 2% by removing false positive detections in unlikely locations.

1. Introduction

Global image appearance carries information about properties of objects in the image, such as their appearance and location. For instance, a picture of a highway taken from a car is more likely to contain cars from the back viewpoint than from the side (fig. 1). A picture of a racing track is more likely to contain racing cars than minivans. This also applies to other classes. A person in a road scene is more likely to be standing than sitting. Another property that can be inferred from global image appearance is the rough location of object instances [3]. For instance, an urban scene with cars parked in front of a building, shows cars in the bottom half of the image (fig. 2).

In this paper we exploit this observation for the benefit of object detection. We propose a method, coined Context Forest (ConF), for learning the relation between the global image appearance and the properties of the objects it contains. Given only the global appearance of a test image, ConF retrieves a subset of training images that contain objects with similar properties. ConF is based on the Ran-

dom Forest [4, 5] framework, which provides high computational efficiency and the ability to learn complex, non-linear relations between global image appearance and objects properties. It is very flexible and only requires these properties to be defined through a distance function between two object instances, e.g. their appearance similarity or difference in location. We demonstrate ConF by learning to predict two properties: aspects of objects appearance and location. ConF trained to predict appearance is then used to speed up multi-component object detectors [1, 2] and ConF trained for object location is used to remove false positives.

Multi-component detectors [1, 2] model the appearance variations within an object class as a mixture of several components. Each component is trained to recognize a particular aspect of objects appearance. For example, cars could have viewpoint components [1], such as front and back views, or subclass components [6], such as taxi, ambulance and minivan. There is growing evidence [7] that the performance of object detectors tends to saturate as the amount of training data increases. We conduct an extensive experiment (sec. 4) on a dataset containing 15x the amount of training data than PASCAL VOC 2012 [8] using two popular multi-component detectors: Deformable Part-based Model [1] (DPM) and the Ensemble of Exemplar SVMs [2] (EE-SVM). Our results show that as the size of training set grows, these detectors can absorb the additional intra-class appearance variations and continue to improve their performance, but only if the amount of components is increased accordingly. This extends the related findings of [7] on single linear SVM HOGs to the DPM and EE-SVM cases. Although increasing the number of components significantly improves performance, it also makes the detectors much slower, as all components need to be run on every test image.

We use ConF to select a subset of model components which is most relevant to a particular test image. We then run only those components, obtaining a speed-up. Our experiments show that ConF delivers a 2x speed-up for DPM [1] and 10x speed-up for EE-SVM [2] without loss of accuracy (sec. 5). Hence, ConF makes large multi-component detectors practical. This is particularly useful

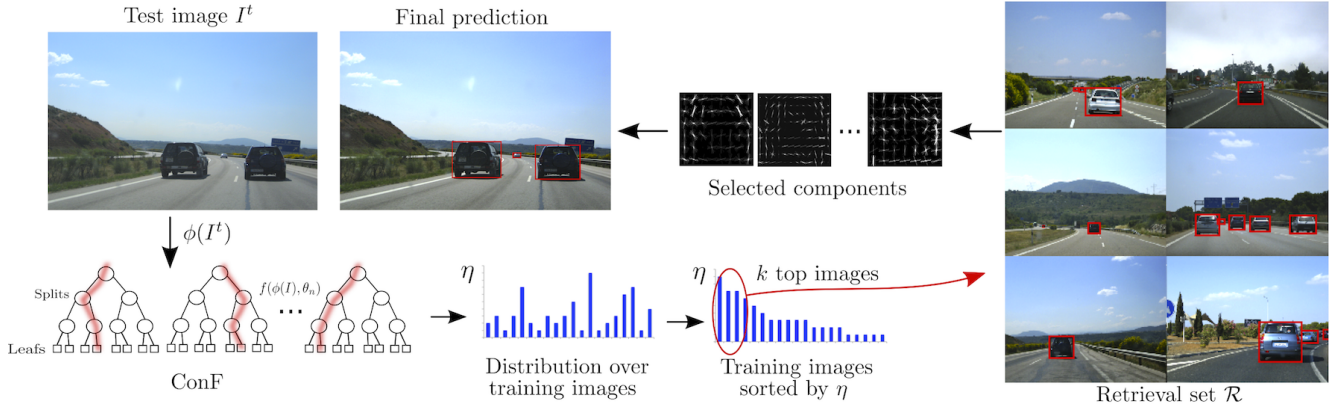


Figure 1: Illustration of ConF selecting components for a test image (sec. 3.1.)

for EE-SVMs, as their number of components is truly very large (i.e. as many as there are training instances). Interestingly, in some cases we even gain a small improvement in accuracy, by not running some components that would produce false positive detections.

Moreover, we train a second ConF to predict at which positions and scales objects are likely to appear in a given test image, analogue to [3]. By incorporating this information in the detector score at test time, we reduce the false positive rate by removing detections in unlikely locations. Experiments show an mAP improvement of 2%. This demonstrates that ConF is a general technique that can predict various kinds of object properties.

Finally, we carry out an extensive comparison to standard nearest-neighbour techniques for such context-based predictions [3, 9–12], which shows that ConF predicts object properties from global image appearance more accurately, it is much faster and more memory efficient (sec. 5).

The rest of the paper is organized as follows. We start by reviewing related work in sec. 2. Sec. 3 explains ConF, our main contribution. In sec. 4 we present a first series of experiments, which study the behaviour of multi-component detectors on large training sets and thus motivate ConF for component selection. Finally, we present a second series of experiments to validate the benefits of ConF on two object detectors in sec. 5.

2. Related work

Context. The use of context for object detection is a broad research area. Some works [11, 13–15] model context as the interactions between multiple object classes in the same image. In this paper, we model context as a relation between global image appearance and properties of the objects within them, as in [3, 9, 10, 12]. These works have shown that global image descriptors give a valuable cue about which classes might be present in an image and where they are located. Since then, many object detectors [1, 16–

19] employed such global context to re-score their detections, thereby removing out-of-context false-positives. A similar approach was taken by [9, 12] for image parsing. All of these works have a nearest neighbour core: they first retrieve a small subset of training images which are most globally similar to a test image, and then transfer the relevant statistics of the object properties in this retrieval set to the test image. In our work instead the retrieval set is estimated by ConF, which is *explicitly trained* to return images containing objects with similar properties to those in the test image. ConF has several advantages over nearest-neighbour approaches: (i) it can learn highly complex non-linear dependencies between the global descriptor and the object property. As a result, it estimates it more accurately; (ii) in large training sets, nearest neighbour becomes very slow, as its complexity is linear in their size. ConF is much faster and more memory efficient; (iii) ConF supports any objective function, which might even be evaluated on a different data representation than the input at test time. This is a crucial feature for our problem, as we want to predict properties of objects, but based on global image features.

Multi-component detectors. These detectors [1, 2, 6, 20–22] model each aspect of an object class as a separate component. They are very popular but can be slow when trained from large training sets as they need many components to reach peak performance. While we present experiments on DPM [1] and EE-SVM [2], ConF can benefit all kind of multi-component detectors, and it allows them to use large training sets without compromising speed.

EE-SVM. The EE-SVM [2] is an extreme case of multi-component detector, where a separate component is created for each training example. Due to that EE-SVM benefit the most from dynamically selecting components with ConF. EE-SVMs are widely used for many applications beyond object detection [23–29]. As they explicitly associate a training example to an object in the test image, they enable

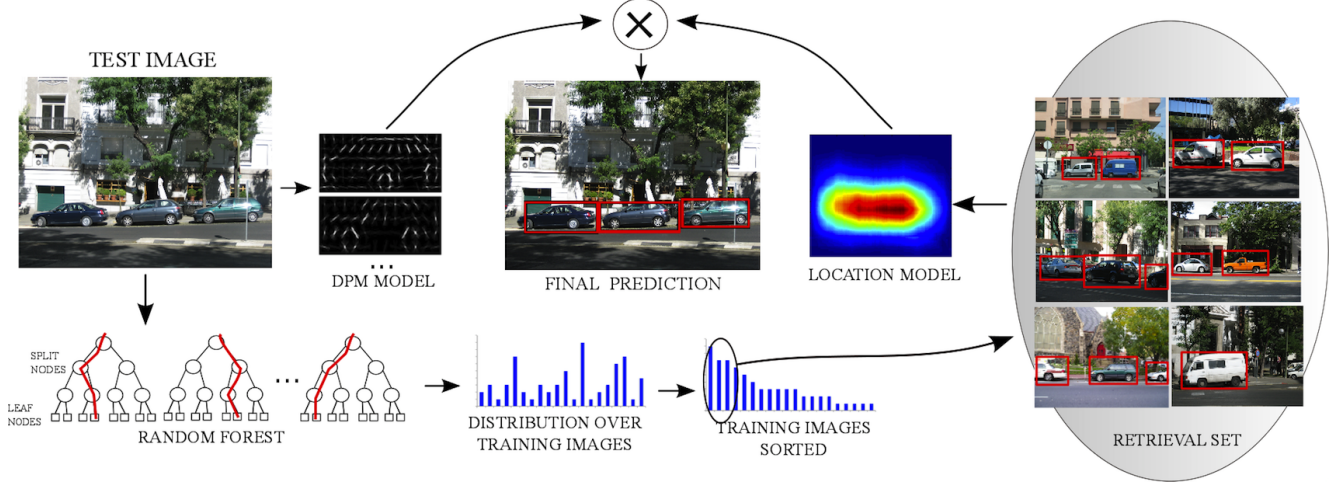


Figure 2: Schematic illustration ConF predicting object locations and their use to improve the DPM score (sec. 3.3).

transferring meta-data such as segmentation masks [2, 27], 3D models [2], subcategory [29] and viewpoint [28]. They can as well be used for discovering mid-level discriminative patches [23, 25, 30, 31] for scene classification [23, 30], for forming a dictionary of object parts [23, 25], or to characterize a certain geo-spatial area [31]. All these applications can potentially be sped-up by ConF.

3. Estimating object properties from context

In this section we exploit the observation that global image appearance contains information about properties of the objects inside it. We focus on two kinds of properties: aspects of appearance and location in the image. We propose a new method, coined Context Forest (ConF), based on the Random Forest framework [4, 5], which learns the relation between global image features and the properties of the object in that image. Given only the global image appearance of a test image, ConF retrieves a subset of training images that contain objects with similar properties.

3.1. Context Forest (ConF)

Given a training set \mathcal{T} the goal of ConF is to map the global appearance $\phi(I_t)$ of a test image I_t into a retrieval set $\mathcal{R} \subset \mathcal{T}$. We want to construct a mapping, such that properties of objects (e.g. appearance, location) in images of \mathcal{R} are similar to the properties of objects in I_t . We describe here our ConF technique in general, then specialize it to the object appearance property in sec. 3.2 and to the object location property in sec. 3.3.

ConF at training time learns an ensemble of decision trees (forest) that operates on global image features $\phi(I)$. We construct each tree by recursively splitting the training set \mathcal{T} at each node. We want the leaves of the trees to contain images whose objects properties are compact according to some measure $c(\mathcal{T}_l)$, where \mathcal{T}_l are the training images in

leaf l . Each internal node n contains a binary split function $f(\phi(I), \theta_n)$, where θ_n are its parameters. Let \mathcal{T}_n be the training images that reached node n , then $f(\phi(I), \theta_n)$ will split \mathcal{T}_n into two subsets \mathcal{T}_l and \mathcal{T}_r . We use axis-aligned weak learners as f [5]. The split function $f(\phi(I), \theta_n)$ applies a threshold to one of the dimensions of image feature vector $\phi(I)$. Following the extremely randomized forest approach [32], for each node we randomly sample several thousand possible splits θ and choose one that maximizes the joint compactness:

$$\theta_n = \arg \max_{\theta} c(\mathcal{T}_l) + c(\mathcal{T}_r) \quad (1)$$

$$\text{s.t. } \forall I \in \mathcal{T}_l, f(\phi(I), \theta) = 0, \forall I \in \mathcal{T}_r, f(\phi(I), \theta) = 1$$

Compactness is defined as

$$c(\mathcal{T}) = \frac{1}{N^2} \frac{1}{\sigma^2 \sqrt{2\pi}} \sum_{w_i \in \mathcal{T}} \sum_{w_j \in \{\mathcal{T} \setminus w_i\}} e^{-\frac{1}{2} \frac{D(w_i, w_j)^2}{\sigma^2}}, \quad (2)$$

where N is the number of ground-truth object bounding-boxes in set \mathcal{T} and $D(w_i, w_j)$ is a distance measure between the properties of two object bounding-boxes w_i and w_j . Note how the inner summation in eq. (2) is an estimation of the density of the distribution induced by all bounding-boxes in $\{\mathcal{T} \setminus w_i\}$, evaluated at w_j . This value is high if w_j has other bounding-boxes nearby. The estimate is done with a Gaussian Kernel Density estimator [33] (KDE). We learn the standard deviation σ from the entire training set once before we train the forest. This determines the scale of the problem, i.e. at which range of distances two bounding-boxes should be considered close. We compute σ as follows. For each training bounding-box w_i we compute its k -nearest neighbours in the whole training set and compute the standard deviation over them. Finally, we set σ as the median of these standard deviations over all bounding-boxes.

By employing different compactness measures c we can use ConF to learn relations between different object properties and global image features. Later we show how to use it for selecting components relevant for a test image (sec. 3.2), for estimating likely object locations in a test image (sec. 3.3).

ConF at test time operates in two phases (see fig. 1 and 2). First, test image I_t is passed through the forest, reaching a leaf in each tree. Thereby, each tree selects the subset of training images contained in that leaf. We now accumulate these selections over all trees in the forest to form the score $\eta(I_i, I_t)$ for $I_i \in \mathcal{T}$, which is the number of trees that have selected I_i . We now construct the retrieval set \mathcal{R} by selecting the k most frequently selected training images. In our experiments $k = 10$.

Note how the split function f and the compactness measure c operate in structurally different spaces. While f operates on the global image features $\phi(I)$, c is measuring the similarity of properties of *objects inside the images*. In this fashion ConF learns the relation between the two. Importantly, c is neither convex nor differentiable in $\phi(I)$ and we are only able to learn this inter-space relation thanks to the unique advantages offered by Random Forests.

3.2. ConF for component selection

Here we assume that each component of an object detector has been previously trained from a visually compact set of object instances, and that we know the component id ξ_j of each training instance j . In EE-SVMs each training instance leads to a unique component. In DPM the component id of a training instance can be inferred by the output of the training procedure [34]. Based on this information, we train a ConF to select a small subset of components to run on a given test image I_t .

Training. To train ConF for components selection we define the distance $D(w_i, w_j)$ in eq. (2) as the L2 distance between the HOG descriptors of object bounding-boxes w_i and w_j .

Test. We pass the test image I_t through ConF obtaining a retrieval set \mathcal{R} . We then estimate a posterior distribution $p(\xi_j|I_t)$ over detector components ξ_j given the test image I_t . As a training image I might contain multiple instances from different components, each training image is ‘labelled’ by a distribution over components $p(\xi_j|I)$. We estimate the component distribution for the test image I_t as the average over the training images in the retrieval set \mathcal{R}

$$p(\xi_j|I_t) = \frac{1}{|\mathcal{R}|} \sum_{I \in \mathcal{R}} p(\xi_j|I) \quad (3)$$

Based on this distribution, we can now select which components to run on I_t . We rank components by their proba-

bility and iteratively pick them until their combined probability mass exceeds a threshold γ . This threshold controls a trade-off between running few components and getting high detection performance. An interesting aspect of our formulation is that the number of selected components changes depending on the test image. A test image with a characteristic appearance matching training images with a systematic recurrence of a few components will lead to a peaky $p(\xi_j|I_t)$. In this case it is safe to run only a few components and we obtain a substantial speedup. On the other hand, if the ConF is uncertain about the contents of the test image, then the entropy of $p(\xi_j|I_t)$ will be high, and many components will be selected. In the extreme case, for a very difficult test image, our procedure naturally degenerates to the default case of running all components.

3.3. ConF for object location

At test time, a typical detector scores hundreds of thousands of windows over the whole test image I_t , based on their appearance only. We propose here to augment the detector’s scores by adding knowledge about likely positions and scales of the object class, derived purely from the global appearance of I_t .

Training. We train two ConFs to predict likely object positions and scales, respectively. To do so, we employ a different measure of compactness, substituting the distance function between two windows in eq. (2) with D_{POS} (or D_{SCALE}). We define $D_{\text{POS}}(w_i, w_j)$ as the L2 distance between the centres of object bounding-boxes w_i and w_j . We define $D_{\text{SCALE}}(w_i, w_j) = \max(\frac{H_i}{H_j}, \frac{H_j}{H_i}) \cdot \max(\frac{W_i}{W_j}, \frac{W_j}{W_i})$ as the difference in their scale (W and H refer to width and height).

Test. At test time, we first pass the test image I_t through ConF obtaining a retrieval set \mathcal{R} , and then compute the following score for each window w in the test image

$$\frac{1}{N} \sum_{w_i \in \mathcal{R}} \frac{1}{\sigma^2 \sqrt{2\pi}} e^{-\frac{1}{2} \frac{D(w, w_i)^2}{\sigma^2}} \quad (4)$$

where N is the number of object instances in the retrieval set \mathcal{R} , and D is either D_{POS} or D_{SCALE} . We learn σ from the entire training set as in sec. 3.1. This score captures how likely a window is to cover an object based on its location or scale. Finally, we linearly combine the location and scale scores with detector’s score of a test window w . The usual non-maxima suppression stage follows.

4. Multi-component detectors on large training sets

In this section we study how multi-component detectors behave when trained on a large training set. We observe that

	Car			Horse		
	+Imgs	Objs	-Imgs	+Imgs	Objs	-Imgs
PASCAL12 [8]	1161	2017	1161	482	710	482
ImageNet [35]	6383	7120	6383	4550	6631	4550
SUN2012 [36]	828	1779	828	-	-	-
Labelme [37]	6566	16743	6566	-	-	-
UIUC [38]	828	889	500	-	-	-
PASCAL-10x [7]	-	-	-	4065	6454	4065
Total	15766	28548	15438	10107	13071	10107
Our training set	14125	25774	13830	9097	12407	9097
Our test set	1641	2774	1608	1010	1388	1010

Table 1: Statistics of the large-scale dataset we assembled by combining images from existing source datasets. The column ‘+Imgs’ reports the number of positive images per dataset; ‘Objs’ is the total number of instances of the class in all positive images; ‘-Imgs’ is the number of negative images we sampled.

it is necessary to increase the number of components as the size of the training set grows, so as to absorb the additional intra-class appearance variation. This motivates using ConF for component selection to speed-up the resulting large mixture models at test time. We perform experiments (sec. 4.3) with two multi-component detectors (DPM [1] and EE-SVM [2], sec. 4.1) on a large-scale dataset (sec. 4.2). A related study was done by [7], using mainly a single component HOG detector ([7], fig. 3-8). Their only experiment with a DPM [1] is on a small training set of 900 faces ([7], fig. 9-10). Hence, our study extends [7] to large-scale training of DPMs and EE-SVMs.

4.1. Multi-component detectors

DPM [1] represents an object class as a collection of parts arranged in a deformable configuration. DPMs use a mixture of components, each specialized to an aspect of the training data to better capture the variation in appearance that the class exhibits. Each component is trained on a subset of the training data with compact appearance, e.g. different viewpoints [1] or subclasses [6]. We use the publicly available implementation [34].

EE-SVM [2] is a model composed of a separate linear SVM classifier for every training instance (exemplar). Each exemplar is represented by a rigid HOG template. The SVM is trained using the exemplar as the only positive against all negatives in the training set. We refer to a single exemplar SVM as a component of the EE-SVM model, by analogy with DPM components. At test time, each component is run on the image independently, producing many candidate detections. These are then filtered by non-maxima suppression in a final stage, where different components compete for the same image region. We use the publicly available implementation [39]. We assemble a large-scale dataset of two classes: car and horse (table 1). Below we discuss the car dataset in detail. The horse dataset was designed analogously.

4.2. Dataset

Source datasets. We combine 6 existing datasets: PASCAL VOC 2012 [8], ImageNet [35], LabelMe [37], SUN 2012 [36], UIUC [38] and PASCAL-10x [7]. PASCAL VOC 2012, PASCAL 10x, and ImageNet contain a variety of images, with both difficult, cluttered images and easier images with big centred cars. UIUC has low resolution, gray-scale images of side-view cars. LabelMe and Sun 2012 contain wide open street scenes with small cars.

Positive images and ground-truth annotations. We collected all images with bounding-box annotations on cars. We took several steps to ensure a clean dataset. First, we removed duplicate images, which were a few hundreds. Next, we removed incorrect bounding-boxes not covering cars or covering a car multiple times. Finally, as some images have unannotated cars, we annotated all missing instances with bounding-boxes for our entire test set (e.g. images from ImageNet). This enables reliable performance measurements.

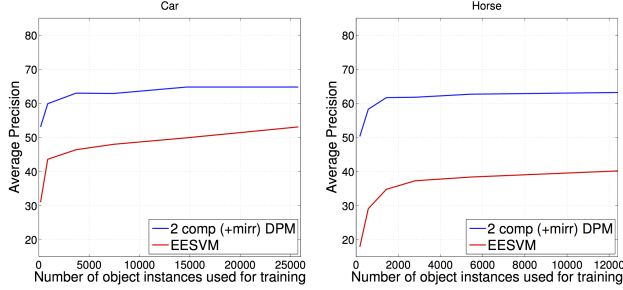
Negative images. We collect negative images from each dataset, so that it contributes an equal number of positive and negative images. To ensure variation, we randomly sampled these negative images.

Train/test splits. We split the dataset into training (90%) and test (10%) sets. Then we split the training set further into 6 increasingly large subsets, which we use in sec. 4.3 to study how the performance of the detectors evolves with increasing training data. We ensure that each split contains images from all source datasets *in the same proportions* to avoid dataset bias issues [40]. These proportions correspond to the percentage of images that each source dataset contributes to the entire dataset (e.g. 8% of all car images are from PASCAL VOC 2012, and 40% from ImageNet).

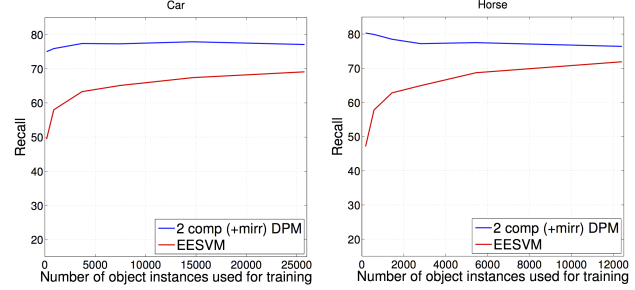
4.3. Experiments

We evaluate the performance of DPM [1] and EE-SVM [2] as a function of the amount of training data and model capacity. First, we analyse how the performance on a fixed test set changes as the amount of training data increases. EE-SVM increase its capacity naturally with each positive example. DPM instead needs its capacity controlled manually. In the second experiment we increase the capacity of the DPM trained on the largest training set, hoping that it will help absorbing the training data better.

Increasing the amount of training data. For each class, we split the training set into 6 nested subsets, which contain 1%, 5%, 10%, 25%, 50% and 100% of the training images (sec. 4.2). Each set is contained in all larger ones, and is composed of images from all source datasets in the same proportions. Fig. 3 shows the average precision (AP) and recall on our test set.



(a) Evolution of AP



(b) Evolution of Recall

Figure 3: Results of experiments when training DPMs and EESVMs with increasing amounts of training data.

For DPM, we use 4 mixture components (2 and their mirrored versions). For both classes, increasing the amount of training data yields a modest improvement in AP. Initially, performance increases roughly logarithmically in the number of training images, but eventually saturates. Surprisingly, this happens quite early. A DPM car detector trained on the whole dataset only performs 1% better than when trained on a third of the data. We observe similar behaviour on horses. Recall, on the other hand, saturates almost immediately for cars and even decreases for horses as the training set grows.

The EE-SVM detector demonstrate continuous, non-saturating growth in both AP and recall as the training data increases. The growth in recall is particularly strong, as it improves by more than 20% for both classes after seeing the full training set.

Increasing DPM capacity. In fig. 4 we help DPM to better absorb the training data by progressively increasing its number of components, while keeping the training set fixed to the largest one. For both classes, performance increases steadily as the number of components grows from 4 (2+2 mirrored) to 16 (8+8) for cars and 10 (5+5) for horses. After that the model starts to overfit: performance decreases and eventually (30 components) drops below the performance of the smallest model (4 components). For cars, each component clearly represents a different aspect (mostly viewpoint).

This overfitting behaviour, even on such a large training set is an interesting finding. The practical implication is that the DPM user has to be careful and manually control the capacity to obtain the best performance. In contrast, EE-SVM does not overfit even when training > 20000 components. Yet, while in terms of growth EE-SVM behaves very well, we note that its absolute detection performance is lower than DPMs.

4.4. Conclusions

In general, both DPM and EE-SVM do benefit from large training sets. The key to continued growth for both meth-

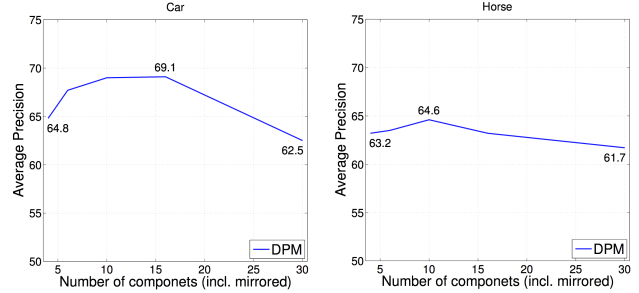


Figure 4: Evolution of AP when increasing the number of DPM components. The performance increases steadily, but eventually it starts overfitting.

ods is control of capacity. EE-SVM automatically increases its capacity, by adding a new component with each training sample. For DPM, the best results are achieved through selecting the right capacity manually, which also corresponds to a rather large number of components (compared to the 2+2 traditionally used on Pascal VOC [8]). Using such large models comes at the cost of longer runtime, as all components have to be applied to a test image. This is especially problematic for EE-SVMs, as running ten thousand components takes about 10 minutes for a single image. In the next section we demonstrate how ConF substantially reduces this computational burden by selecting only a small subset of components most relevant to a particular test image and run only those.

5. Experiments with ConF

In this section we evaluate the performance of ConF for component selection and location estimation. As global image descriptors $\phi(I)$ we extract SURF [41], LAB and SIFT [42] descriptors on a dense grid at multiple scales. For each feature type we train a class-specific codebook of 1000 visual words and construct a 2-level spatial pyramid [43]. Additionally, we also extract a GIST [44] descriptor for the image. Overall, we train ConF on a 16000 dimensional feature space, using 750 trees for each task.

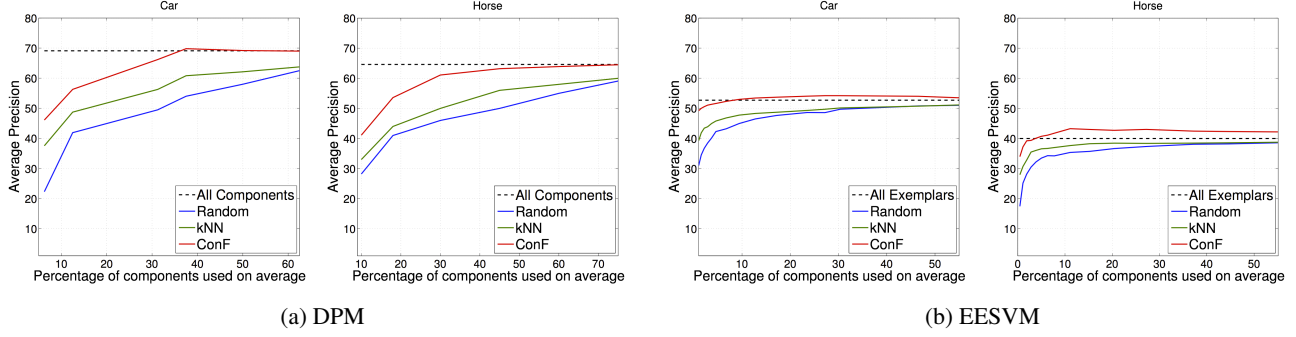


Figure 5: Results of applying ConF for the automatic component selection. The points on the plot correspond to different choices for the threshold γ (sec. 3.2). The horizontal axis corresponds to the average amount of components used. The vertical axis corresponds to the AP of the detector using components selected by ConF.

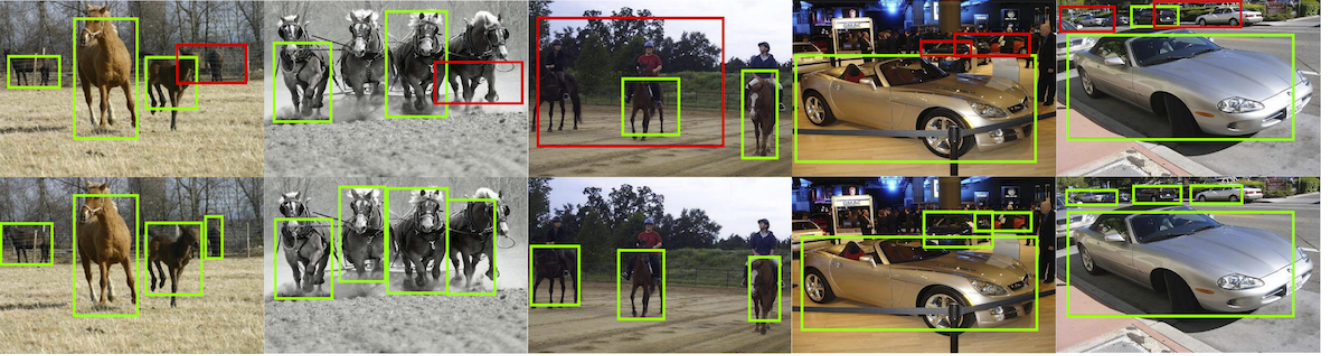


Figure 6: Detection obtained before (top row) and after (bottom row) applying ConF for component selection. Green bounding-boxes highlight correct detections, while red ones show false positives.

Quality of retrieval sets. We quantify how similar object bounding-boxes from a test image I_t are to those in the retrieval set \mathcal{R} returned by the method as follows

$$\frac{1}{Z\sigma^2\sqrt{2\pi}} \sum_{w_i \in I_t} \sum_{w_j \in \mathcal{R}(I_t)} e^{-\frac{1}{2} \frac{D(w_i, w_j)^2}{\sigma^2}}, \quad (5)$$

where Z is number of pairs of bounding-boxes in I_t and \mathcal{R} . The distance D and standard deviation σ vary depending on the property (appearance, position, scale) as defined in sec. 3.2, 3.3.

Table 2 show results averaged over the test set, higher values are better. As a baseline, we return the whole training set as the retrieval set. This leads to a generic prior on image properties, independent of the test image. Moreover, we compare to the traditional way of building retrieval sets by k-nearest neighbours (kNN) [3, 9, 10, 12, 27], defined on the same features as ConF. Both kNN and ConF greatly outperform the baseline, proving they return meaningful retrieval sets. This confirms the observation that the global image appearance conveys information about the objects properties inside the images. Moreover, ConF returns better retrieval sets than kNN across all object properties and retrieval set sizes evaluated.

Automatic component selection. We now use ConF to select object detector components relevant for a given test image. We use the best performing settings from sec. 4.3, i.e. 16 (10) component DPM for cars (horses) and *very large* EE-SVMs using all training exemplars, i.e. 25774 for cars and 12407 for horses.

Fig. 5 shows the evolution of AP while increasing the percentage of components used (higher is better). We compare to building retrieval sets by kNN, and to a baseline which randomly selects components without looking at the test image. ConF outperforms the baseline and kNN for both object classes, for both detection models, and over the whole range of the plots. By employing ConF, we closely match the performance of the full DPM model by running roughly half of the components. We match the performance of a full EE-SVM when running less than 10% of the components. Even in the extreme case of running just *one* EE-SVM component, the AP is about 90% of that of the full model. Interestingly, for EE-SVM on the horse class, ConF *improves AP by 3%* over the full ensemble using all components, when running *10× fewer components*. There is also a minor improvement in AP for EE-SVM on the car class, when running half of the components. The AP improve-

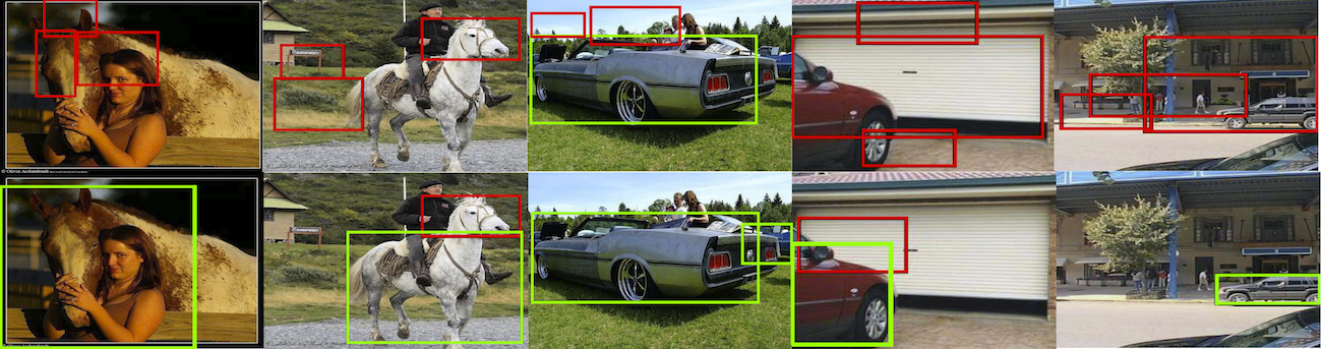


Figure 7: Detection obtained before (top row) and after (bottom row) applying ConF as location model. Green bounding-boxes highlight correct detections, while red ones show false positives.

Obj property \ Data	Car					Horse				
	All train data	NN		ConF		All train data	NN		ConF	
		1	10	1	10		1	10	1	10
Appearance	0.01	0.10	0.03	0.12	0.09	0.03	0.04	0.04	0.06	0.05
Position	0.36	1.30	0.88	1.84	1.42	0.37	0.50	0.53	0.57	0.57
Scale	0.03	0.05	0.04	0.08	0.07	0.06	0.08	0.08	0.09	0.08

Table 2: Evaluation of the quality of retrieval sets for predicting object properties. Each entry represents the average density of the retrieval set evaluated at the objects properties in the test images. We consider two sizes for the retrieval set $|\mathcal{R}| = 1$ and $|\mathcal{R}| = 10$.

Car					
DPM			EE-SVM		
None	NN	ConF	None	NN	ConF
69.1	0	+2.1	52.7	0	+2.3

Horse					
DPM			EE-SVM		
None	NN	ConF	None	NN	ConF
64.6	0	+0.7	40	0	+1.1

Table 3: The results of augmenting the detector score with the location model derived by ConF (sec. 3.3) and NN compare to not using location model at all (None).

ment comes from dropping some components that lead to false positives.

These experiments demonstrate the ability of ConF to select relevant components given just global image appearance. This makes EE-SVMs practical even when trained from large sets with tens of thousands of exemplars (the average runtime for a test image decreases from 10 minutes to 1 minute on our machine with 4 i5-core 3.10GHz processor and 16 GB memory). Fig. 6 shows some example results.

Object locations. Here we demonstrate how ConF trained to estimate the location of objects from global image features can improve detection performance by downgrading the score of false positives at unlikely locations. As tab. 3 shows, this improves AP for both classes and both detectors (+2% for cars and +1% for horses). Instead, kNN

does not bring any improvement, further confirming that ConF returns better retrieval sets. Fig. 7 shows example results.

Computational and memory efficiency. ConF does not only offer better performance than kNN, but is also more memory and computationally efficient. In terms of computation, kNN requires a number distance computations linear in the number of training images, where ConF requires only a logarithmic number of threshold operations.

In terms of memory, kNN stores all feature vectors of all images in the training set. For cars, this amounts to 1.68 GB. For each internal node ConF stores a threshold, a feature id and the ids of its children, amounting to 16 bytes. The leaves store the indices of the training images they contain, for a total of exactly the number of training images $\times 2$ bytes overall (per tree). For cars, there are < 900 internal nodes on average per tree. As we store 750 trees per class, the grand total is only 27 MB ($60\times$ less than kNN).

Conclusions. We propose a novel method — ConF, which learns the relation between the global image appearance and properties of objects in the image. We show how ConF can be employed to dynamically select a small number of components for a test image. This improves speed and, sometimes, even detection performance. We have also shown how to use ConF to predict likely object location to reduce false positive rates.

References

- [1] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part based models," *IEEE Trans. on PAMI*, vol. 32, no. 9, 2010.
- [2] T. Malisiewicz, A. Gupta, and A. A. Efros, "Ensemble of exemplar-svms for object detection and beyond," in *ICCV*, 2011.
- [3] B. Russell, A. Torralba, C. Liu, R. Ferugs, and W. Freeman, "Object recognition by scene alignment," in *NIPS*, 2007.
- [4] L. Breiman, "Random forests," *ML Journal*, vol. 45, no. 1, pp. 5–32, 2001.
- [5] A. Criminisi, J. Shotton, and E. Konukoglu, "Decision forests for classification, regression, density estimation, manifold learning and semi-supervised learning," *Microsoft Research Cambridge, Tech. Rep. MSRTR-2011-114*, 2011.
- [6] S. Divvala, A. Efros, and M. Hebert, "How important are 'deformable parts' in the deformable parts model?," in *ECCV*, 2012.
- [7] X. Zhu, C. Vondrick, D. Ramanan, and C. Fowlkes, "Do we need more training data or better models for object detection?," in *BMVC*, 2012.
- [8] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results." <http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html>.
- [9] C. Liu, J. Yuen, and A. Torralba, "Nonparametric scene parsing: Label transfer via dense scene alignment," in *CVPR*, 2009.
- [10] A. Torralba, "Contextual priming for object detection," *IJCV*, vol. 53, no. 2, pp. 153–167, 2003.
- [11] A. Rabinovich, A. Vedaldi, C. Galleguillos, E. Wiewiora, and S. Belongie, "Objects in context," in *ICCV*, 2007.
- [12] J. Tighe and S. Lazebnik, "Superparsing: Scalable nonparametric image parsing with superpixels," in *ECCV*, 2010.
- [13] G. Heitz and D. Koller, "Learning spatial context: Using stuff to find things," in *ECCV*, 2008.
- [14] M. Choi, J. Lim, A. Torralba, and A. Willsky, "Exploiting hierarchical context on a large database of object categories," in *CVPR*, 2010.
- [15] C. Desai, D. Ramanan, and C. Folkess, "Discriminative models for multi-class object layout," in *ICCV*, 2009.
- [16] K. Murphy, A. Torralba, and W. T. Freeman, "Using the forest to see the trees: A graphical model relating features, objects, and scenes," in *NIPS*, 2003.
- [17] H. Harzallah, F. Jurie, and C. Schmid, "Combining efficient object localization and image classification," in *ICCV*, 2009.
- [18] J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, and A. W. M. Smeulders, "Selective search for object recognition," *IJCV*, vol. 104, no. 2, pp. 154–171, 2013.
- [19] "University of amsterdam and euvision technologies at ilsrvrc2013 (ILSVRC)." <http://www.image-net.org/challenges/LSVRC/2013/slides/ILSVRC2013-UvA-Euvision-web.pdf>, 2013.
- [20] C. Gu, P. Arbeláez, Y. Lin, K. Yu, and J. Malik, "Multi-component models for object detection," in *ECCV*, 2012.
- [21] B. Drayer and T. Brox, "Training deformable object models for human detection based on alignment and clustering," in *ECCV*, 2014.
- [22] O. Aghazadeh, H. Azizpour, J. Sullivan, and S. Carlsson, "Mixture component identification and learning for visual recognition," in *ECCV*, 2012.
- [23] S. Singh, A. Gupta, and A. A. Efros, "Unsupervised discovery of mid-level discriminative patches," in *ECCV*, 2012.
- [24] Y. Aytar and A. Zisserman, "Enhancing exemplar svms using part level transfer regularization," in *BMVC*, 2012.
- [25] I. Endres, K. J. Shih, J. Jiaa, and D. Hoiem, "Learning collections of part models for object recognition," in *CVPR*, 2013.
- [26] J. Dong, W. Xia, Q. Chen, J. Feng, Z. Huang, and S. Yan, "Subcategory-aware object classification," in *CVPR*, 2013.
- [27] J. Tighe and S. Lazebnik, "Finding things: Image parsing with regions and per-exemplar detectors," in *CVPR*, 2013.
- [28] C. Gu and X. Ren, "Discriminative mixture-of-templates for viewpoint classification," in *ECCV*, 2010.
- [29] V. Jain and E. Learned-Miller, "Online domain adaptation of a pre-trained cascade of classifiers," in *CVPR*, 2011.
- [30] M. Juneja, A. Vedaldi, C. Jawahar, and A. Zisserman, "Blocks that shout: Distinctive parts for scene classification," in *CVPR*, 2013.
- [31] C. Doersch, S. Singh, A. Gupta, J. Sivic, and A. A. Efros, "What makes paris look like paris?," in *SIGGRAPH*, 2012.
- [32] F. Moosman, B. Triggs, and F. Jurie, "Fast discriminative visual codebook using randomized clustering forests," in *NIPS*, 2006.
- [33] E. Parzen, "On the estimation of a probability density function," *Annals of Mathematical Statistics*, vol. 33, no. 3, pp. 1065–1076, 1962.
- [34] R. B. Girshick, P. F. Felzenszwalb, and D. McAllester, "Discriminatively trained deformable part models, release 5." <http://people.cs.uchicago.edu/~rbg/latent-release5/>, 2012.
- [35] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-fei, "ImageNet: A large-scale hierarchical image database," in *CVPR*, 2009.
- [36] J. Xiao, J. Hays, K. Ehinger, A. Oliva, and A. Torralba, "SUN database: Large-scale scene recognition from Abbey to Zoo," in *CVPR*, 2010.
- [37] B. C. Russel and A. Torralba, "LabelMe: a database and web-based tool for image annotation," *IJCV*, vol. 77, no. 1-3, pp. 157–173, 2008.
- [38] S. Agarwal, A. Awan, and D. Roth, "Learning to detect objects in images via a sparse, part-based representation," *IEEE Trans. on PAMI*, vol. 20, no. 11, pp. 1475–1490, 2004.
- [39] T. Malisiewicz, "Ensemble of e-svms implementation." <https://github.com/quantombone/exemplarsvm>, 2011.
- [40] A. Torralba and A. Efros, "An unbiased look on dataset bias," in *CVPR*, 2011.
- [41] H. Bay, A. Ess, T. Tuytelaars, and L. van Gool, "SURF: Speeded up robust features," *CVIU*, 2008.
- [42] D. Lowe, "Distinctive image features from scale-invariant keypoints," *IJCV*, vol. 60, no. 2, pp. 91–110, 2004.
- [43] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories," in *CVPR*, 2006.
- [44] A. Oliva and A. Torralba, "Modeling the shape of the scene: a holistic representation of the spatial envelope," *IJCV*, vol. 42, no. 3, pp. 145–175, 2001.